

REMARKS

Reconsideration of this application as amended is respectfully requested. In the Office Action claims 1-51 are pending. Claims 1-51 are rejected. Claims 1-19, 27-32, and 40-51 are objected to. In this response, claims 2, 12, 19, 23, 34, 44 and 51 have been cancelled. Claims 1, 8, 13-15, 20, 27, 30-33, 35-43, and 45-50 have been amended. No new matter has been added.

I. CLAIM OBJECTIONS

Claims 1-19, 27-32, and 40-51 have been objected to because of informalities specified in the Office Action such as poor sentence structure and typographical errors. In view of the foregoing amendments, it is respectfully submitted that the claim objections have been overcome. Applicant submits that the aforementioned amendments relating to informalities specified in the Office Action are non-narrowing because the amendments do nothing more than resolve formatting and typographical errors.

With respect to claim 27, the Office Action states that the claim recites poor sentence structure in line 11 where it recites “logic to prune walking of nodes.” [Office Action, Sept. 28, 2006, page 9]. Applicant has reviewed the sentence structure and respectfully disagrees. The claim language reciting “logic to prune walking of nodes” does not recite poor sentence structure. The word “prune” here is used in its secondary meaning as a verb “to cut off unwanted parts (as of a tree).” See Merriam-Webster’s Collegiate Dictionary, 11th Edition. Applicant believes the Office Action has confused the word “prune” to be a noun and, thus, has erroneously determined that claim 27 recites poor sentence structure. Thus, Applicant is using the phrase “logic to prune walking of nodes” correctly and claim 27 does not recite poor

sentence structure. Accordingly, Applicant respectfully requests the claim objection be withdrawn.

II. REJECTIONS UNDER 35 U.S.C. § 112, SECOND PARAGRAPH

The Office Action has rejected claim 32 under 35 U.S.C. § 112, second paragraph, as being indefinite for insufficient antecedent basis. Applicant has amended claim 31 from “a set of trees” to “a plurality of storage trees.” Applicant submits that the amendment is non-narrowing because a “set of trees” is the same as a multiple trees. Applicant supports this position by pointing to the Office Action which equates a “set of trees” with more than one tree or multiple trees. See Office Action, p. 10 (“[i]f the set from Claim 32 is *one tree*, Claim 31’s set of trees, would make Claim 31 indefinite.”). Thus, Applicant submits that amending Claim 31 from “a set of trees” to “a plurality of storage trees” is a non-narrowing amendment.

Claim 32 has been amended from “the set of trees” to “the plurality of storage trees” to remove the insufficient antecedent basis. Additionally, the “set of one or more storage trees” has been amended to “one or more of the plurality of storage trees.” Applicant submits that the foregoing amendment is non-narrowing for the same reasons discussed above.

III. REJECTIONS UNDER 35 U.S.C. § 101

The Office Action has rejected claims 1-19 and 31-51 under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. [Office Action, pp. 10-11]. Applicant respectfully submits that the amendments to independent claims 1, 8, 15, 31, 33, 40 and 47 overcome the rejections.

Specifically for claims 1-7, 15-19, 33-39 and 47-51 the Office Action asserts these claims recite no practical application. [Office Action, p. 11]. Applicant has amended claims 1, 15, 33 and 47 to include the limitation of performing “garbage collection in a storage device.” Applicant submits that the foregoing amendments now recite a practical application, and that being to make un-referenced memory space in a storage device available for subsequent programs or processes.

Specifically for claims 1-7, 8-14, 15-19, 33-39, 40-46 and 47-51 the Office Action states that these claims recite no tangible result. Id. As a result, claims 1-7, 8-14, 15-19, 33-39, 40-46 and 47-51 have been amended to further clarify the tangible result, and that being a performing garbage collection in a storage device. Garbage collection operations are performed within storage systems to delete data that is no longer referenced/active so as to reclaim unused memory for subsequent programs or processes.

Specifically for claims 33-39, 40-46, and 47-51 the Office Action states that these claims are not limited to tangible embodiments. Applicant has amended claims 33-39, 40-46, and 47-51. As a result, claims 33-39, 40-46, and 47-51 are now specifically directed to a “machine-storage medium that stores instructions.” Applicant submits the amendment now places the foregoing claims in a condition for allowance because a machine-storage medium that stores instructions is a tangible embodiment. Machine storage media includes things such as read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; and etc. [See spec., amended paragraph 33].

Specifically for claims 31 and 32, the Office Action asserts these claims fail because claim 31 appears to be no more than a program *per se*. Id. Applicant questions whether this may have been a typographical error. For clarification, in claim 31 it is the apparatus that

Applicant is claiming which, by definition, is more than a program *per se*. The claimed apparatus includes a backup system which is made up of the following components: 1) tracking logic; 2) allocator logic; and 3) a garbage collection logic. Applicant further supports this argument by pointing to the specification which indicates that the present invention may be implemented by hardware or software or a combination of hardware and software. See spec., paragraph [0034] (“[a]lternatively, the features or operations of the present invention are performed by specific hardware components which contain hard-wired logic for performing the operations, or by any combination of programmed data processing components and specific hardware components. Embodiments of the present invention include software, data processing hardware, data processing system-implemented methods, and various processing operations, further described herein.”). As a result, Applicant submits that a system complying with claim 31 is more than a program *per se*.

In view of the foregoing amendments, it is respectfully submitted that the rejections have been overcome. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

IV. REJECTIONS UNDER 35 U.S.C. § 103 OVER ZWILLING IN VIEW OF HITZ

The Office Action has rejected claims 1-7, 18, 20-39, and 50 under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,249,792 to Zwilling, et al., (“Zwilling”) in view of U.S. Patent No. 5,963,962 to Hitz, et al. (“Hitz”). Applicant does not admit that Hitz and Zwilling are prior art and reserves the right to swear behind either reference at a later date. It is respectfully submitted that Hitz and Zwilling fail to disclose or suggest the limitations set forth above, individually or in combination.

a. Overview of Zwilling

Zwilling discloses an on-line dynamic file shrink facility composed of shrinking log files. See id. at p. 14. Zwilling involves a method of shrinking a file's size without unloading the file, deleting and recreating the file, and reloading the data. See Zwilling at col. 1, lines 36-39. To do this, Zwilling discloses a shrink facility that works hand-in-hand with a file allocation manager to shrink files by reclaiming "empty space." The data records of each file are scanned to determine the amount of data in the file based on the number of used allocation units or data blocks within the file. The amount of free space in the file, if any, is estimated and a fence address is determined based on this estimate. See Zwilling at col. 5, lines 13-22. The estimated fence address represents the place in the file where the empty space would begin if it was all collected at the end of the file. For example, if the amount of free space is estimated at "x" bytes, then the fence address will be located "x" bytes from the tail of the file. The "allocation units" in the file are then scanned starting at the tail of the file until the fence address is reached. See id. at col. 5, lines 29-34. If an "allocation unit" above the fence (between the tail of the file and the fence) contains data, the file allocation manager locates and allocates an empty unit below the fence and copies the data into the allocated unit below the fence (between the fence and the head of the file), and the unit above the fence is de-allocated. See id. at col. 5, lines 34-43. When the fence address is reached, all the blocks above the fence will have been de-allocated and the process is complete. Id. Zwilling relies on data already having been deleted, and, therefore, is not identifying what can be deleted, just what can be de-allocated.

b. Overview of Hitz

Hitz is directed to maintaining the consistency of the file system as it changes over time to avoid data loss due to system crashes. Hitz provides a method for maintaining a file system in a consistent state and for creating read-only copies (snapshots). See Hitz at cols. 3-4, lines 66-1. Each snapshot uses no disk space when it is initially created. Id. at col. 4, lines 14-15. As time progresses, the active file system and a particular snapshot share fewer and fewer data blocks. Id. at col. 18, lines 8-12. Therefore, the snapshot file space begins to grow. The Hitz system cleans up its file system by deleting snapshots when they “consume unacceptable numbers of disk blocks.” Id. at col. 18, lines 12-14. The snapshots become unacceptably large, and, hence, in need of deleting, because the snapshots are pointing to blocks of data that are no longer in use by the active file system.

In addition to the deleting snapshots, the blocks of data no longer referenced by any remaining snapshot should also be deleted. To do this, the blocks of data associated with the snapshots are managed using a block map (blkmap) file. Id. at lines 29-32. Each block of data in the Hitz system has this blkmap file which contains a 32-bit entry. Id. at col. 4, lines 31-33. Bit 0 indicates whether the block is within the active file system and the next 20 bits are used for tracking whether the block resides in up to 20 snapshots. Id. at col. 4, lines 32-35. Whenever a block of data is in the active file system, bit 0 is set, and whenever a snapshot is taken of the data block, the snapshot bits 1-20 are set depending on which of the 20 available snapshots the block of data resides in. Id. at col. 4, lines 32-34. As old snapshots are deleted, the corresponding bits are reset to zero. When all 20 bits are reset to zero, then the data block is no longer referenced by any of the 20 snapshots. Id. at col. 20, lines 27-35. Similarly, when Bit 0 is reset to 0, the block is no longer within the active file system. Id. At the point where all bits are zero, the block is no longer referenced by a snapshot and may be deleted. Id.

Thus, to locate what blocks of data can be deleted, Hitz relies on the 32-bit entries in the blkmap file.

The Hitz file system also saves disk space by only duplicating the root node (referred to as the “root inode”) that describes the inode file. This is in contrast to other prior art systems that duplicate the entire inode file. The inode file is much larger than the root inode because the inode file is used to describe the root directory of a file system. Id. at col. 4, lines 18-20. So, in Hitz, the duplicated inode is enough to describe the inode file which, in turn, describes the root directory which describes the rest of the file system. Id. at col. 11, lines 1-4 (“[b]ecause the root inode 1610B and 1612B ... describes the inode file 1620, that in turn describes the rest of the files 1630-1660 in the file system 1670 including all meta-data files 1630-1640 ...”). “[T]he root inode 1610B and 1612B is viewed as the root of a tree of blocks” and the “WAFL system 1670 uses this tree structure for its update method (consistency point) and for implementing snapshots.” Id. at lines 4-8.

c. Independent Claims 1, 20 and 33

- i. Applicant finds insufficient guidance in the Office Action on how the references are to be combined to properly respond.

The Office Action states, “[i]t would have been obvious to one having ordinary skill in the art at the time of invention having the teachings of Hitz and Zwilling before him/her to take the write anywhere file-system layout from Hitz and install it into the invention of Zwilling, thereby offering the obvious advantage of taking snapshots of garbage collected data.” [Office Action, p. 14]. Applicant is having trouble understanding what this means. Applicant submits that the Office Action does not provide enough detail on how to it proposes to “install” Hitz into the system described by Zwilling. If the rejection in the Office

Action is maintained, Applicant respectfully requests a more detailed explanation of how the references are being combined.

Applicant submits that the Office Action is proposing one of three alternative combinations: 1) using Hitz to take snapshots on Zwilling's garbage collected data; 2) using Zwilling to shrink files *before* taking snapshots using Hitz; or 3) using Zwilling to shrink files *after* taking snapshots using Hitz.

ii. Alternative # 1: Using Hitz to take snapshots on Zwilling's garbage collected data

Using Hitz to take snapshots of Zwilling's garbage collected data blocks is one way to read the combination in the Office Action. The Office Action states, "[i]t would have been obvious to one having ordinary skill in the art at the time of invention having the teachings of Hitz and Zwilling before him/her to take the write anywhere file-system layout from Hitz and install it into the invention of Zwilling, thereby offering the obvious advantage of taking snapshots of garbage collected data." [Office Action, p. 14]. Applicant submits that it would not make sense to use Hitz to take snapshots on Zwilling's garbage collected data. The term "garbage collected data" indicates that collection has already been performed. If the data is garbage, then it should be scheduled for deletion. Taking a snapshot of garbage data would have the result of reproducing data that is garbage. This is contrary to the purpose of both Zwilling and Hitz which seek to conserve space. Also, this is contrary to the assertion in the Office Action that makes the combination for "saving space." See Id. at p. 14. As a result, Applicant submits that a person having ordinary skill in the computer sciences art would not have combined the Hitz and Zwilling references in the manner indicated by the Office Action. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

iii. Alternative # 2: Using Zwilling to shrink files *before* taking snapshots using Hitz

Using Zwilling to shrink files *before* applying Hitz to create snapshot trees of data blocks having smaller file sizes is another way to read the combination in the Office Action.

In this configuration, the data is compressed before the snapshot of Hitz is taken so that the space taken up by the blocks of data in each snapshot becomes smaller. As a result, the overall size of each snapshot is reduced, “thus saving space.” [Office Action, p. 14].

Applicant submits that a person having ordinary skill in the art with the Zwilling and Hitz references in front of him/her would build the system of combination alternative #2. This is because a person having skill in the art would recognize the advantages similar to those asserted in the Office Action, and that being, to take snapshots of data that has already been compressed using Zwilling. Applicant submits, therefore, that alternative #2 would result in a system that is most likely capable of “offering the obvious advantage of taking snapshots of garbage collected data, thus saving space.” [Office Action, p. 14]. However, such a system requires using Zwilling to shrink files *before* taking snapshots using Hitz. While the combination makes the most sense as a combination, the order of this combination is backward in relation to what is claimed.

Applicant has amended claims 1, 20 and 33 to further distinguish the cited art. Claim 1 now recites as follows:

1. A method of garbage collecting in a storage device comprising:
locating blocks of data in a log that are referenced and within a range at a tail of the log using pruned walking, the range representing an address range within an allocated segment of the log,

wherein the log is implemented in a hierarchical architecture having a plurality of storage trees, each storage tree representing a snapshot taken at a point in time of target data being processed, each storage tree having a plurality of nodes and each node representing a segment block of data of a snapshot associated with each storage tree; ~~and~~

copying the blocks of data that are referenced by one or more other blocks of data of other nodes and within the range to an unallocated segment of the log, wherein blocks of data that are not referenced by other blocks of data and within the range remain untouched, and

marking the range at the tail of the log as unallocated so that at least a portion of an address space within the range can be reclaimed.

Claims 20 and 33 have been similarly amended.

Applicant submits that even if combination alternative #2 is made, this combination would not read on claims 1, 20 and 33 (as amended). Claims 1, 20 and 33 explicitly require “garbage collecting in an storage device” which comprises at least the following: 1) “locating blocks of data ... in a hierarchical architecture having a plurality of storage trees;” 2) “copying the blocks of data that are referenced ... to an unallocated segment;” and 3) “marking the range at the tail of the log as unallocated so that at least a portion of an address space within the range can be reclaimed.” Claims 1, 20 and 33 expressly require garbage collecting and garbage collecting is deciding which data blocks within a storage device can be deleted and then deleting them. However, combination alternative #2 is the opposite of this. Alternative #2 requires applying Zwilling to shrink the files *before* applying Hitz to take snapshots of the shrunk files. Thus, alternative #2 is working on the data blocks before they are present in the snapshots taken by Hitz. Therefore, alternative #2 cannot read on claims 1, 20 and 33 as amended. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

iv. Alternative # 3: Using Zwilling to shrink files *after* taking snapshots using Hitz

Using Zwilling to shrink files *after* applying Hitz to create the snapshot trees is another way to read the combination in the Office Action. Zwilling requires blocks of data within a file to be empty (already deleted since Zwilling does not locate blocks to be deleted) so that it

can shrink files. Zwilling, col. 1, lines 21-31. Thus, under combination alternative #3 the combination would be to use Hitz to determine which data blocks are unreferenced/inactive based on the 32-bit entries, delete those blocks, and then use Zwilling to shrink what remains (because Zwilling works by shrinking already deleted blocks).

However, amended claims 1, 20 and 33 require performing garbage collection (locating referenced/active blocks and copying out of the range before the data within them is deleted) by walking the storage trees representing the blocks of data. Applicant respectfully submits that neither Hitz nor Zwilling disclose walking the storage trees to determine referenced/active blocks of data. Hitz doesn't walk storage trees because Hitz uses the 32-bit entries in the blkmap file. Also, Hitz locates the **unreferenced/inactive** blocks as opposed to **referenced/active** blocks as claimed, which is another distinction. In other words, Hitz uses a 32-bit entry so that Hitz does not have to walk each of the data blocks to figure out which are unreferenced/inactive so that it can delete them. See Hitz col. 9, lines 48-50 (“[a] block is available as a free block in the file system when all bits (BIT0-BIT31) in the 32-bit entry 110A for the block are clear.”). See also Id. at lines 61-64 (“[i]f bit 0 (BIT0) is set to a value of 0, this does not necessarily indicate that the block is available for allocation. All the snapshot bits must also be zero for the block to be allocated.”). This does not read on claims 1, 20 and 33 as amended because the claims require determining whether the blocks are referenced/active by walking the storage trees (using “pruned walking”). Hitz does not walk the storage trees because it is unnecessary. All Hitz needs to do is check to see if all the bits in the 32-bit entry are cleared. So, Hitz does not disclose walking of storage trees as required by the claims. Zwilling also fails to disclose walking of storage trees as required by the claims because Zwilling does not determine which blocks are referenced/active. As a result, Zwilling cannot determine which blocks of data to delete (i.e., what is not still part of at least

one snapshot). Rather, Zwilling scans for empty blocks (already deleted), not unreferenced/inactive blocks. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

Thus, the combination does not read on claims 1, 20 and 33 because Zwilling and Hitz, individually or in combination, do not disclose any of the following as required by the claims: 1) performing “garbage collecting in a storage device;” 2) locate blocks “that are referenced” by other blocks; and 3) walk the data blocks “using pruned walking.” As a result, Applicant submits that a person having ordinary skill in the computer sciences art would not have combined the Hitz and Zwilling references in the manner indicated by the Office Action. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

Further, it is not clear how the pointers in Hitz are being updated when Zwilling moves the data around. Specifically, the combination of Hitz and Zwilling does not disclose how the root inode, the root inode file, blkmap file, inomap file, or the plurality of inodes for the files in the rest of the file system would be updated when Zwilling is used to relocate the blocks. As a result, Applicant submits that a person having ordinary skill in the computer sciences art would not have combined the Hitz and Zwilling references in the manner indicated by the Office Action. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

- v. A person having ordinary skill in the art would not have been motivated to combine the Hitz and Zwilling references in the way combination alternative #3 has.

Applicant submits that the combination that makes the most sense is combination alternative #2. As discussed above, alternative #2 would result in a system that is most likely capable of “offering the obvious advantage of taking snapshots of garbage collected data, thus saving space.” [Office Action, p. 14]. However, such a system requires using Zwilling to

shrink files *before* taking snapshots using Hitz. As discussed above, such a system would be the reverse of what Applicant is claiming. See supra. So, even assuming arguendo that a person having ordinary skill in the art would have been motivated to combine the references under alternative #2, such a combination would be backwards from what Applicant is claiming.

On the other hand, Applicant submits that a person having ordinary skill in the art would *not* have been motivated to combine the Hitz and Zwilling references as indicated in combination alternative #3. Applicant respectfully submits that alternative #3 is a forced combination in an attempt to make it fit the claims. This is because the Hitz system would not have benefited from the advantages of the Zwilling system in alternative #3 and vice versa. The references are solving two distinctly different problems. Zwilling is shrinking files to remove empty space and Hitz is maintaining consistency in a file system that is changing over time.

The differences between these two systems are significant. The Hitz system is not trying to reduce the size of individual files by reclaiming empty space. Hitz is directed to maintaining the consistency of the file system as it changes over time to avoid data loss due to system crashes. Hitz, col. 18, lines 8-12. Shrinking these files does not help solve Hitz's problem because it doesn't matter how large or small these files are; the point is that they aren't being referenced by any other data, and, therefore, in need of deletion.

The Zwilling system, on the other hand, is not concerned with how consistent the file system is. All Zwilling is doing is solving the problem of how to shrink individual files while users are accessing the online file system. Creating snapshots of files and tracking them over time in order to maintain consistency in a file system does not teach or suggest a solution to the problem of reducing file sizes in an online file system without disrupting users. Zwilling

operates to reclaim empty space within individual files. To do this, Zwilling sets a fence address and moves all allocation units that are not empty below the fence and de-allocates the space above the fence. Whether or not Hitz is deleting unreferenced files from the system does not have anything to do with the size of the files.

Therefore, there is no motivation to combine the references in the way alternative #3 has combined them because Zwilling's shrinking files does not help Hitz solve the problem of maintaining file consistency over time, and Hitz's maintaining file consistency over time does not help Zwilling shrink file sizes to reclaim unused space. So, neither reference, either expressly or impliedly, provides motivation to combine the references in the way alternative #3 has. One having ordinary skill in the art faced with either of the problems solved by the Hitz and Zwilling references would not have been motivated to use the other reference to help solve the problem.

Further, the Office Action states, "[i]t would have been obvious to one having ordinary skill in the art at the time of invention having the teachings of Hitz and Zwilling before him/her to take the write anywhere file-system layout from Hitz and install it into the invention of Zwilling, thereby offering the obvious advantage of taking snapshots of garbage collected data." [Office Action, p. 14]. The Office Action seems to be saying that the Zwilling shrinking of files would work on the Hitz snapshot trees to reduce space as the number of snapshots increases. However, if this were the case, then you would want to install Zwilling in Hitz and not vice versa. It would not make sense to install Hitz into Zwilling for the same reason it would not make sense to install a file manager of an entire file system into a single file. Zwilling operates on single files, whereas Hitz manages an entire file system. Hitz is managing data blocks which contain multiple files.

- vi. The exemplary advantages of a system that complies with claims 1, 20 and 33 render it nonobvious over the cited references.

A system that complies with claims 1, 20, and 33 solves the problem of performing garbage collection across a large memory space without having to investigate each node to determine if it is referenced by other nodes of other storage trees. By way of illustration, one exemplary use of the claimed method is for backup storage systems. Such a backup system stores snapshots of a target system's local memory over time. This may be accomplished by copying the target's entire local memory systematically at some pre-determined interval. The target data is then stored in a storage device within the backup system. Over time, this results in large memory requirements for the backup storage system. Such a system must be capable of storing voluminous data on the order of a terabyte, for example. For large memory structures, pruned walking of the storage trees is more efficient than walking all of the storage trees, and does not require storage of the 32-bit vector for every block as required in Hitz. Thus, one reason a system that complies with the claims above is nonobvious over the cited reference is that such a system performs "pruned walking" of the data structure to save time and memory space required for tracking.

Additionally, a log records activity that takes place over time. For example, the log of a backup system contains information about what files are backed up and the times when they are backed up. The head of a log, therefore, is where all the new data is placed as time goes forward. Accordingly, a system that complies with claims 1, 20, and 33 designates the tail of the log as the currently selected range to be cleaned because the tail of the log will contain the oldest data, and the oldest data will most likely contain the most unreferenced/inactive blocks. The system that complies with claims 1, 20 and 33 accomplishes garbage collection by designating a range of addresses at the tail of the log and walking only those blocks to see if

any are still referenced/active. If they are, then they are copied to the head of the log (where all the new data is located) and the range of addresses that was selected to be clean is de-allocated to reclaim unused memory for subsequent programs or processes.

As a result, the exemplary advantages of a system that complies with the claims 1, 20 and 33 render the present application nonobvious over the cited references. Accordingly, Applicant requests withdrawal of the rejections.

d. **Independent claims 27 and 31**

Applicant also submits that the exemplary advantages render the system that complies with claims 27 and/or 31 nonobvious over the cited references according to the above discussion with respect to claims 1, 20, and 33.

i. **Using Zwilling to shrink files *after* creating snapshots using Hitz**

The Office Action asserts that “[i]t would have been obvious to one of ordinary skill in the art at the time of invention to take the storage device (making 2 storage devices) and the storage trees of snapshots from Hitz and install it into the system of Zwilling, thereby offering the obvious advantage of extending Zwilling’s invention to work on archived (snapshot) files in attempts to save space as the snapshot size increases thereby increasing the number of active snapshots in Hitz.” [Office Action, p. 22]. Applicant submits that the resulting system would not be the system that results from using Zwilling to shrink files after using Hitz to create the snapshot trees. Applicant submits that such a system would not be one that offers “the obvious advantage of extending Zwilling’s invention to work on archived (snapshot) files in attempts to save space as the snapshot size increases thereby increasing the number of active snapshots in Hitz” for the same reasons as those articulated above with respect to

combination alternative #3. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

e. **Dependent Claims 2-7, 21-26, 28-30, 32 and 34-39**

Given that the rest of the claims are dependent on one of the above independent claims, either directly or indirectly; for reasons similar to those discussed above, it is respectfully submitted that the rest of the claims are also patentable over Zwilling and/or Hitz. Thus, withdrawal of these rejections is also respectfully requested.

V. **REJECTIONS UNDER 35 U.S.C. § 103 OVER ZWILLING**

The Office Action has rejected claims 8-17, 19, 40-49, and 51 under 35 U.S.C. §103(a) as being unpatentable over Zwilling. It is respectfully submitted that claims 8-17, 19, 40-49, and 51 include limitations that are not disclosed or suggested by Zwilling.

a. **Independent Claims 8 and 40**

With respect to claims 8 and 40 Applicant has made amendments to overcome the rejections. Specifically, Applicant has amended claim 8 as follows:

A method comprising:

garbage collecting within a range of addresses in a storage system having a plurality of storage trees, each storage tree having a plurality of nodes and having multiple references to the same block of data, the garbage collecting including:

pruning walking of the plurality of storage trees to determine active blocks of data within said range, where active blocks of data are those still in one of the plurality of storage trees, the pruning walking including:

determining, based on accessing in one of said plurality of storage trees a parent node that has a plurality of descendent nodes, that none of the plurality of descendant nodes are associated with blocks of data within the range; and

skipping the walking of the plurality of descendent nodes based on said determining,

wherein the active blocks determined to be in the range are copied out of the range and the range is marked as unallocated so that at least a portion of the address space within the range can be reclaimed.

Claim 40 has been similarly amended. The Office Action asserts that Zwilling teaches the (now amended) limitation of “garbage collecting within a range of addresses in a storage system ...” Applicant respectfully disagrees. First, the Zwilling reference is directed to an online dynamic shrink facility to reduce the size of files, whereas claims 8 and 40 each require garbage collection operations within a range of addresses in a storage system. Nowhere in the Zwilling reference is either “a storage system” or “garbage collection operations over a range of addresses” disclosed.

Applicant also submits that the combination of the two different embodiments in Zwilling as indicated by the Office Action would not perform garbage collection in the manner required by the claims. Garbage collection is a term that would be defined by a person having ordinary skill in the computer sciences art as a form of automatic memory management. Garbage collection operations are performed within storage systems to delete data that is no longer referenced/active so as to reclaim unused memory for subsequent programs or processes. Zwilling does not teach or determine whether an allocation unit is unreferenced/inactive or not. Rather, Zwilling is concerned with shrinking files in which data has been deleted and copying the data blocks within those files based on whether they have data in them. Even if a file was no longer referenced/active in a file system, Zwilling could be performed to shrink the data in that file. So, Zwilling does not copy or delete allocation

units based on whether or not they are referenced/active. As such, Zwilling does not perform garbage collection. Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

The Office Action states that Zwilling discloses a storage system having “a plurality of storage trees.” [Office Action, p. 27]. The Office Action argues that the binary tree disclosed in Zwilling is multiple storage trees. [Office Action, p. 3]. However, even assuming, arguendo, that the binary tree discloses a “storage” tree as required by the claims, the Applicant is claiming a “plurality of storage trees.” Zwilling does not teach or suggest a plurality of storage trees because Zwilling does not teach or suggest a plurality of binary trees. Rather, Zwilling discloses a *single* binary tree “in which the pages containing the data records form leaf nodes of the tree.” Zwilling, col. 8, lines 49-51. Further, the “[n]odes above the leaf nodes in the binary tree comprise pages containing index records and are referred to as intermediate nodes, with the topmost node called the root node.” *Id.* at lines 51-54. Applicant submits that the binary tree disclosed in Zwilling, therefore, is a single centralized tree used for indexing stored data records across different pages of a file. In addition, the Office Action later admits that Zwilling does not disclose a “plurality of storage trees.” [Office Action, p. 13 (“Zwilling discloses the above limitations but does not expressly teach: ... having a plurality of storage trees ...”). Therefore, Applicant submits that Zwilling fails to disclose a storage system having “a plurality of storage trees” as required by the claims. Accordingly, Applicant respectfully requests withdrawal of the claim rejection.

The Office Action goes on to argue that “a tree in its most basic form is one node and since Zwilling at least implies multiple nodes in the binary tree, this easily makes multiple storage trees. [Office Action, p. 3]. Applicant respectfully disagrees. The Office Action states that a tree in its most basic form is a node and multiple nodes make up multiple trees.

Applicant asserts that this is somewhat nonsensical. Assuming, arguendo, that a tree in its most basic form is a node, then it logically follows that multiple nodes would add up to a tree and not add up to multiple trees as asserted by the Office Action. Moreover, amended claims 8 and 40 now expressly require “a plurality of storage trees each storage tree having a plurality of nodes ...” Applicant respectfully submits that the amendment of claims 8 and 40 further differentiates the Zwilling reference because the claims now explicitly require a plurality of storage trees each having multiple nodes which is not disclosed in Zwilling.

Thus, Applicant submits that the binary tree in Zwilling does not teach or suggest “a plurality of storage trees.” Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

Additionally, the Office Action asserts that Zwilling teaches “having multiple references to the same block.” [Office Action, p. 3]. Applicant respectfully disagrees. Zwilling discloses only one binary tree, not multiple trees. Since the Zwilling only discloses a single tree, there can be no references between more than one tree as with a plurality of trees. Claims 8 and 40 expressly require multiple storage trees having references to the same block of data. This being the case, it follows that there must be references between storage trees to reference the same block among a plurality of storage trees. Once again, since Zwilling only discloses one tree, it cannot have references between trees. The Office Action argues that “[m]ultiple references to the same block (page) are the data records in the page with their doubly-linked nature. [Office Action, p. 3]. However, the double linkage disclosed in Zwilling is between different sibling nodes and between sibling and parent nodes of the same binary tree. Here, the same arguments apply because a multiple reference to the same block from a plurality of storage trees requires cross-tree references. Since Zwilling only discloses one tree, the reference does not disclose cross-tree references. Therefore, Applicant

submits that Zwilling also fails to disclose “having multiple references to the same block.”

Accordingly, Applicant requests withdrawal of the claim rejections.

The Office Action asserts that Zwilling discloses (indirectly) “pruning walking of the plurality of storage trees to determine active blocks of data within said range.” [Office Action, p. 4]. A system that complies with claims 8 and 40 provides for a pruned (or abbreviated) search of the blocks of data that are referenced (or active) within a segment to be cleaned within a log. However, Zwilling does not perform walking of the storage trees for several reasons. As discussed above, Zwilling does not determine which blocks are referenced/active. As a result, Zwilling cannot determine which blocks of data to delete. Rather, Zwilling scans the binary tree for empty blocks (already deleted), and not unreferenced/active blocks.

Additionally, the Office Action points the traversal of the binary tree in Zwilling as performing pruned “walking of the plurality of storage trees.” As discussed above, Zwilling does not include “a plurality of storage trees,” and, logically, cannot disclose the walking of that which it does not disclose. As such, Zwilling cannot perform pruned walking of a plurality of storage trees. Therefore, Zwilling fails to disclose “pruning walking” of a plurality of storage trees for at least this reason.

Zwilling also fails to disclose the limitation of “skipping walking of the plurality of descendant nodes based on said determining.” Since Zwilling does not walk the storage trees, Zwilling does not determine whether there are any referenced/active nodes. As a result, Zwilling also cannot determine if there are NOT any referenced/active nodes, and does not determine if there are NOT any descendant nodes within the range. It follows, then, that Zwilling does not skip the walking of these nodes because Zwilling cannot skip the walking of nodes based on a determination that was never made. In other words, because Zwilling

fails to determine that there are NO referenced/active descendant nodes within the range, it cannot then skip the walking of these nodes because it cannot tell which nodes to walk and which to skip. In fact, Zwilling expressly does not skip the walking of nodes because the reference describes scanning the entire file to determine if each allocation is used or empty. See Zwilling at col. 5, lines 13-15 (“[w]hen the file 205 is to be reduced in size, the shrink facility 201 scans the file 205 from its start, or head, to its end, or tail”). Zwilling does not skip the walking of certain nodes because Zwilling always scans the entire file and never abbreviates this process. Therefore, Applicant submits that Zwilling also fails to disclose “skipping walking of the plurality of descendant nodes.” Accordingly, Applicant respectfully requests withdrawal of the claim rejections.

b. Independent Claims 15 and 47

Regarding claims 15 and 47, Applicant has amended the claims to further distinguish from the cited references as follows:

15. A method of garbage collecting in a storage device comprising:

performing the following operations until each block of data that is active in a range to be cleaned at a tail of a log of data is copied to a head of the log, wherein the range to be cleaned is a range of addresses in a storage system having a plurality of storage trees each storage tree having a plurality of nodes, wherein a block of data is associated with a node of a storage tree, each tree having a plurality of nodes, the operations including:

copying blocks of data associated with child nodes of a current node that are within the range to be cleaned to the head of the log;

retrieving a block of data associated with the current node, upon determining that a minimum address value among addresses of descendent nodes is within the range to be cleaned;

designating, as the current node, one of the child nodes of the current node that is an interior node, upon determining that at least one child node is an interior node; ~~and~~

designating, as the current node, an ancestor node of the current node whose descendent nodes are unprocessed; and

marking the range as unallocated when the blocks of data that are active and within the range are copied to the head of the log so that at least a portion of the address space within the range to be cleaned can be reclaimed.

Claim 47 has been similarly amended.

Applicant submits that both claims 15 and 47, as amended, require at least the following limitations: 1) “garbage collecting in a storage device;” and 2) “having a plurality of storage trees each storage tree having a plurality of nodes.” which, as discussed above, are not disclosed in the Zwilling reference. Accordingly, Applicant submits that the same arguments as above with respect to claims 8 and 40 apply equally to claims 15 and 47. Therefore, Applicant respectfully requests withdrawal of the claim rejections.

c. **Dependent Claims 9, 10, 11, 13, 14, 41, 42, 43, 45, and 46**

Because each of claims 9, 10, 11, 13, 14, 41, 42, 43, 45, and 46 depend on one of claims 8 and 40, either directly or indirectly, Applicant respectfully requests withdrawal of the claim rejections for at least the same reasons as given above with respect to claims 8 and 40.

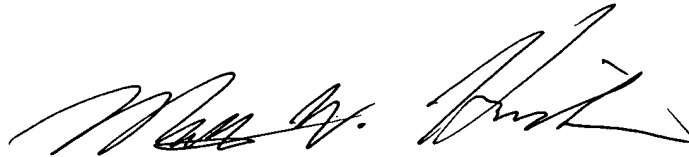
VI. CONCLUSION

In view of the foregoing, Applicant respectfully submits the present application is now in condition for allowance. If the Examiner believes a telephone conference would expedite or assist in the allowance of the present application, the Examiner is invited to call the undersigned attorney at (408) 720-8300. Please charge Deposit Account No. 02-2666 for any shortage of fees in connection with this response.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

Date: February 14, 2006

A handwritten signature in black ink, appearing to read 'Matthew W. Hindman', written over a horizontal line.

Matthew W. Hindman
Reg. No. 57,396
matthew_hindman@bstz.com

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026
(408) 720-8300